

# Technical Cybersecurity

objdump

# GNU Binutils

CONTAINS LOTS OF USEFUL TOOLS

---

- ▶ objdump
- ▶ readelf
- ▶ strings
- ▶ nm
- ▶ strip
- ▶ ...and many more

# Sample C Program

WHAT IS THIS?

---

- ▶ A simple program
- ▶ Calls two functions
- ▶ We'll use to example stack management, registers, etc.

```
function2.c
1
2 void call2(void) {
3     int k = 0xcafebabe;
4 }
5
6 void call(void) {
7     int j = 0xcafed00d;
8     call2();
9 }
10
11 int main(int argc, char* argv[])
12     int i = 0xdead0de;
13     call();
14     return 0;
15 }
16
```

# Sample makefile

## MAKEFILES USE TABS

---

- ▶ Remember, you need to *tab* in from target names
- ▶ This will compile the function2.c file
- ▶ We'll add flags
- ▶ **\$ make**

```
makefile
28  ASM=exit.asm
29  OBJ=function2.o
30  CC_FLAGS=-g
31
32  %.o: %.c
33      $(CC) -c -o $@ $< $(CC_FLAGS)
34
35  main: $(OBJ)
36      $(CC) -o f2 function2.o $(CC_FLAGS)
37
38  clean:
39      rm *.o f2
40      rm -rf a.out
41
42
```

# objdump

## THE EXECUTABLE

---

▸ **\$ objdump**

## DISASSEMBLY

---

▸ **\$ objdump -d f2 > f2.dis**

**See the hexspeak?**

```
2 5f4: 5d                pop    %rbp
3 5f5: e9 66 ff ff ff    jmpq   560 <register_tm_clc
4
5 00000000000005fa <call2>:
6 5fa: 55                push   %rbp
7 5fb: 48 89 e5          mov     %rsp,%rbp
8 5fe: c7 45 fc be ba fe ca  movl    $0xcafebabe,-0x4(%rb
9 605: 90                nop
0 606: 5d                pop     %rbp
1 607: c3                retq
2
3 0000000000000608 <call>:
4 608: 55                push   %rbp
5 609: 48 89 e5          mov     %rsp,%rbp
6 60c: 48 83 ec 10       sub     $0x10,%rsp
7 610: c7 45 fc 0d d0 fe ca  movl    $0xcafed00d,-0x4(%rb
8 617: e8 de ff ff ff    callq  5fa <call2>
9 61c: 90                nop
0 61d: c9                leaveq %rbp
1 61e: c3                retq
2
3 000000000000061f <main>:
4 61f: 55                push   %rbp
5 620: 48 89 e5          mov     %rsp,%rbp
6 623: 48 83 ec 20       sub     $0x20,%rsp
7 627: 89 7d ec          mov     %edi,-0x14(%rbp)
8 62a: 48 89 75 e0       mov     %rsi,-0x20(%rbp)
9 62e: c7 45 fc de c0 ad de  movl    $0xdeadc0de,-0x4(%rb
0 635: e8 ce ff ff ff    callq  608 <call>
1 63a: b8 00 00 00 00    mov     $0x0,%eax
2 63f: c9                leaveq %rbp
3 640: c3                retq
4 641: 66 2e 0f 1f 84 00 00  nopw    %cs:0x0(%rax,%rax,1)
5 648: 00 00 00
6 64b: 0f 1f 44 00 00    nopl    0x0(%rax,%rax,1)
7
8 0000000000000650 <__libc_csu_init>:
9 650: 41 57             push    %r15
0 652: 41 56             push    %r14
1 654: 49 89 d7          mov     %rdx,%r15
2 657: 41 55             push    %r13
3 659: 41 54             push    %r12
4 65b: 4c 8d 25 8e 07 20 00  lea     0x20078e(%rip),%r12
5 662: 55                push    %rbp
6 663: 48 8d 2d 8e 07 20 00  lea     0x20078e(%rip),%rbp
```

```

12 5f4: 5d      pop    %rbp
13 5f5: e9 66 ff ff jmpq   560 <register_tm_clon
14
15 000000000000005fa <call2>:
16 5fa: 55      push   %rbp
17 5fb: 48 89 e5 mov    %rsp,%rbp
18 5fe: c7 45 fc be ba fe ca movl   $0xcafebabe,-0x4(%rbp)
19 605: 90      nop
20 606: 5d      pop    %rbp
21 607: c3      retq
22
23 00000000000000608 <call>:
24 608: 55      push   %rbp
25 609: 48 89 e5 mov    %rsp,%rbp
26 60c: 48 83 ec 10 sub    $0x10,%rsp
27 610: c7 45 fc 0d d0 fe ca movl   $0xcafed00d,-0x4(%rbp)
28 617: e8 de ff ff callq  5fa <call2>
29 61c: 90      nop
30 61d: c9      leaveq
31 61e: c3      retq
32
33 0000000000000061f <main>:
34 61f: 55      push   %rbp
35 620: 48 89 e5 mov    %rsp,%rbp
36 623: 48 83 ec 20 sub    $0x20,%rsp
37 627: 89 7d ec mov    %edi,-0x14(%rbp)
38 62a: 48 89 75 e0 mov    %rsi,-0x20(%rbp)
39 62e: c7 45 fc de c0 ad de movl   $0xdeadc0de,-0x4(%rbp)
40 635: e8 ce ff ff callq  608 <call>
41 63a: b8 00 00 00 00 mov    $0x0,%eax
42 63f: c9      leaveq
43 640: c3      retq
44 641: 66 2e 0f 1f 84 00 00 nopw   %cs:0x0(%rax,%rax,1)
45 648: 00 00 00
46 64b: 0f 1f 44 00 00 nopl   0x0(%rax,%rax,1)
47
48 00000000000000650 <__libc_csu_init>:
49 650: 41 57    push   %r15
50 652: 41 56    push   %r14
51 654: 49 89 d7 mov    %rdx,%r15
52 657: 41 55    push   %r13
53 659: 41 54    push   %r12
54 65b: 4c 8d 25 8e 07 20 00 lea     0x20078e(%rip),%r12
55 662: 55      push   %rbp
56 663: 48 8d 2d 8e 07 20 00 lea     0x20078e(%rip),%rbp

```

```

28 ASM=exit.asm
29 OBJ=function2.o
30 CC_FLAGS=-g
31
32 %.o: %.c
33     $(CC) -c -o $@ $< $(CC_FLAGS)
34
35 main: $(OBJ)
36     $(CC) -o f2 function2.o $(CC_FLAGS)
37
38 clean:
39     rm *.o f2
40     rm -rf a.out
41
42

```

```

2 void call2(void) {
3     int k = 0xcafebabe;
4 }
5
6 void call(void) {
7     int j = 0xcafed00d;
8     call2();
9 }
10
11 int main(int argc, char* argv[]) {
12     int i = 0xdeadc0de;
13     call();
14     return 0;
15 }

```

# Other options

## MIX SOURCE & ASM

---

- **\$ objdump -S f2 | less**

## LOAD FLAGS

---

- **\$ objdump -f f2**

```
0000000000000608 <call>:
void call(void) {
608: 55                push    %rbp
609: 48 89 e5          mov     %rsp,%rbp
60c: 48 83 ec 10       sub     $0x10,%rsp
        int j = 0xcafed00d;
610: c7 45 fc 0d d0 fe ca  movl    $0xcafed00d,%edi
        call2();
617: e8 de ff ff ff    callq   5fa <call2>
}
61c: 90                nop
61d: c9                leaveq  %rbp
61e: c3                retq

000000000000061f <main>:
int main(int argc, char* argv[]) {
61f: 55                push    %rbp
620: 48 89 e5          mov     %rsp,%rbp
623: 48 83 ec 20       sub     $0x20,%rsp
627: 89 7d ec          mov     %edi,-0x14(%rbp)
62a: 48 89 75 e0       mov     %rsi,-0x20(%rbp)
        int i = 0xdeadcode;
62e: c7 45 fc de c0 ad de  movl    $0xdeadcode,%edi
        call();
635: e8 ce ff ff ff    callq   608 <call>
        return 0;
63a: b8 00 00 00 00    mov     $0x0,%eax
}
63f: c9                leaveq  %rbp
640: c3                retq
641: 66 2e 0f 1f 84 00 00  nopw    %cs:0x0(%rax,%r
648: 00 00 00
64b: 0f 1f 44 00 00    nopl    0x0(%rax,%r
```

More Binutils coming  
up!